

AD-A052 023

STANFORD UNIV CALIF DIGITAL SYSTEMS LAB
EMMYXL USER'S GUIDE. (U)
MAR 76 W A WALLACH

UNCLASSIFIED

DSL-TN-84

ARO-12958.5-M

F/6 9/2

DAAG26-76-6-0001

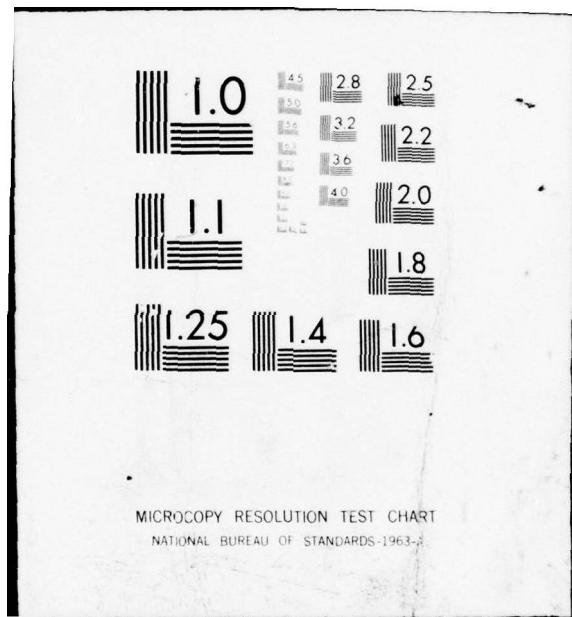
NL

| OF |
AD A052 023



END
DATE
FILED
5-78

DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA052023

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE			READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER	2. GOVT ACCESSION NO.		3. RECIPIENT'S CATALOGUE NUMBER	
Technical Note No. 84/2958.5-m			DSL-TN-84	
4. TITLE (and Subtitle)	5. TYPE OF REPORT & PERIOD COVERED			
EMMYXL USER'S GUIDE	Technical Note			
6. AUTHOR(s)	7. PERFORMING ORG. REPORT NUMBER			
Walter A. Wallach	DAAG-26-76-G-0001			
8. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS			
Digital Systems Laboratory Stanford Electronics Laboratories Stanford University, Stanford, CA 94305	12 J4Pp.1			
9. CONTROLLING OFFICE NAME AND ADDRESS	11. REPORT DATE			
U.S. Army Research Office-Durham	March 1976			
12. NUMBER OF PAGES	13. SECURITY CLASS. (of this report)			
36				
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. DECLASSIFICATION/DOWNGRADING SCHEDULE			
16. DISTRIBUTION STATEMENT (of this Report)	Approved for public release; distribution unlimited.			
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)				
18. SUPPLEMENTARY NOTES	<p>The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.</p> <p style="text-align: right;">CH</p>			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)				
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)	<p>This document is intended as a guide to the use of EMMYXL, the expression-oriented line-by-line assembler developed by Hedges for the Stanford Emulation Lab. It is intended to be used along with the Principles of Operation for the Stanford EMMY (TN #65, Dec., 1975) and the EMMY/360 Assembler (TN #74, Dec. 1975). Various IBM OS/370 and VSII documents may also prove useful.</p> <p style="text-align: center;">NH</p>			

408 071-*Hue*

EMMYXL USER'S GUIDE

by

Walter A. Wallach

March 1976

Technical Note No. 84

Digital Systems Laboratory
Stanford Electronics Laboratories
Stanford University
Stanford, California

The work herein was supported in part by the Army Research Office-Durham
under Grant No. DAAG29-76-G-0001.

ACCESSION for	White Section
NTIS	Buff Section
DOC	UNANNOUNCED
CLASSIFICATION	
DISTRIBUTION/AVAILABILITY COPIES	
A	

Digital Systems Laboratory
Stanford Electronics Laboratories

Technical Note No. 84

March 1976

EMMYXL USER'S GUIDE

by

Walter A. Wallach

ABSTRACT

This document is intended as a guide to the use of EMMYXL, the expression-oriented line-by-line assembler developed by Hedges for the Stanford Emulation Lab. It is intended to be used along with the Principles of Operation for the Stanford EMMY (TN # 65, Dec., 1975) and the EMMY/360 Assembler (TN #74, Dec. 1975). Various IBM OS/370 and VSII documents may also prove useful.

The work herein was supported in part by the Army Research Office-Durham under contract DAAG29-76-G-0001.

Table of Contents

1.1	EMMYXL -----	1
1.2	Expressions -----	1
1.3	Multiply Step -----	3
1.4	Divide Step -----	6
1.5	Branching and Conditional -----	10
1.51	T-Machine Conditional -----	11
1.52	A-Machine Conditional -----	11
2.0	Crossassembler Operation -----	14
2.1	Titles and Comments -----	14
2.2	Location Counter Control -----	14
2.3	Listing Options -----	15
2.4	Code Generation -----	16
3.0	EMMY Simulator -----	18
	References and Related Material -----	19
	Appendix A - Error Flags -----	20
	Appendix B - Using EMMYXL -----	23
	Appendix C - Sample EMMYXL Listings -----	26
	Multiply and Divide	
	Appendix D - Sample Object Code -----	31
	and Object File Format	
	Appendix E - Sample Simulation Run -----	33
	and Instruction Trace	
	Appendix F - Control Card Summary -----	36

1.1 EMMYXL

EMMYXL is an expression oriented, line-by-line assembler language which provides a concise, straightforward expression for each possible EMMY operation. The two sides of each microinstruction are coded by separate expressions delimited by a semicolon (";"). Conditional expressions are enclosed entirely in parenthesis (that is, a CONDITIONAL T-op would be coded by enclosing the entire instruction, both T- and A-side, in parenthesis. A conditional A-op branch would be coded by enclosing only the A-side in parenthesis). See TN#74.

The syntax of the EMMYXL language is presented in a separate document (TN#74) and will not be repeated here. Some points will be clarified and discussed.

1.2 Expressions

An expression is a sequence of identifiers and symbols separated by compile time operators (double operators, such as ++ or --). Expressions are evaluated left to right. More than two terms in an expression may lead to unpredictable results. Note, since expressions are evaluated right to left, the expression:

8 - 3 + 4

is evaluated to $8-(3+4)=1$, rather than $(8-3)+4=9$.

Expressions may appear as arguments of DC and EQU pseudo ops.

See TN#74 for a more detailed discussion.

1.3 Multiply Step

Operation of the EMMY multiply step is described here. Sample code for a 32 bit multiply is included in the Appendix.

The multiply step is coded as:

MUS(AF,BF)

where AF and BF are host register identifiers. A single multiply step proceeds as follows:

- 1) The double length value obtained by concatenating the AF register contents (placed in bits <63:32>) and AF \oplus 1 contents (placed in bits <31:0>) form the PRODUCT. Thus, the AF operand specifies an even/odd register pair. Either the even or the odd register may be specified, and it becomes the high order 32 bits of the PRODUCT. The other register becomes the low order 32 bits. (e.g., if AF is specified as 3, the product is REG[3] | REG[2]). (Note: the symbol \oplus denotes EXCLUSIVE OR).
- 2) The least significant bit of the product is saved, and the product shifted right arithmetically by one bit. If OVERFLOW was set (by the previous MUS), then the sign bit is inverted.

- 3) If the least significant bit of the product was one (before the shift), then the multiplier, obtained from REG[BF], is added to the high 32 bits of the product. OVERFLOW is set or reset as needed. If no addition is performed, OVERFLOW is reset.
- 4) PRODUCT<63:32> is returned to REG[AF] and PRODUCT<31:0> returned to REG[AF \oplus 1].

Since the arithmetic shift precedes the addition of the multiplier, an extra alignment step is necessary to properly align the 64 bit result. Thus, when multiplying two 32 bit numbers, 33 multiply steps are required, or 32 multiply steps followed by a right double arithmetic shift.

Programming Considerations

- 1) Operands may be either positive or negative two's complement values.
- 2) When doing arithmetic on quantities of fewer than 32 bits, post shifting of the result may be required.
- 3) The register specified as AF must be cleared to zero before the first MUS is executed. Register AF \oplus 1 must contain the

multiplicand. Values should not be modified between multiply steps.

- 4) After the required number of multiply steps, REG[AF] | REG[AF ⊕ 1] contain the 64 bit binary product.

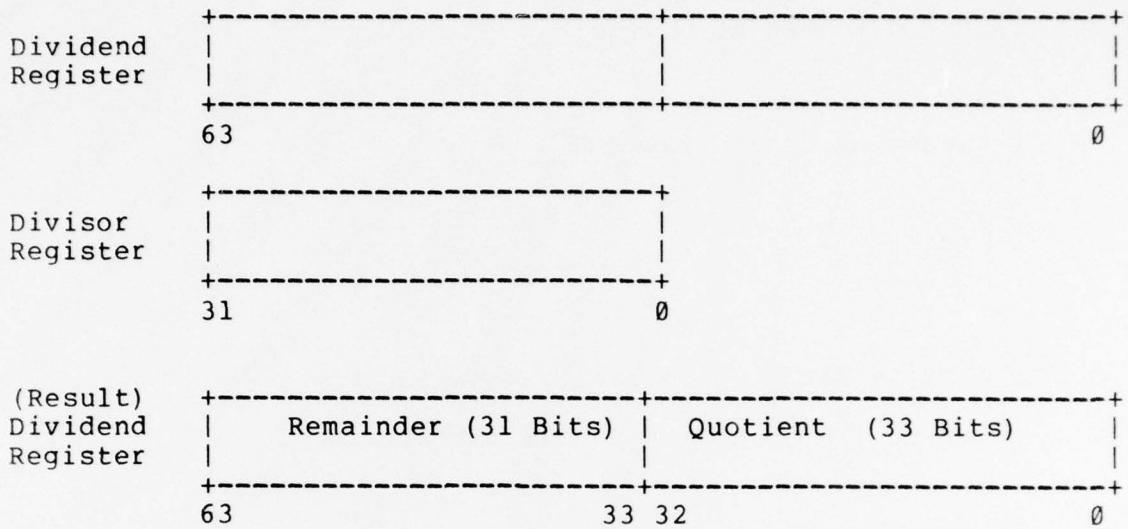
1.4 Divide Step

Operation of the EMMY Divide Step is described. Sample code for a 32 bit binary divide is included in the Appendix.

The EMMY Divide Step uses a restoring division algorithm to accomplish binary division of a 64 bit dividend by a 32 bit divisor. The result after 33 divide steps is a 33 bit quotient and a 31 bit remainder.

The algorithm proceeds in much the same way as ordinary long division. The divisor is subtracted from the high-order 32 bits of the dividend. If the result is ≥ 0 , it replaces the high-order 32 bits of the dividend. The dividend is then shifted left by one bit, shifting in a low-order one if the subtraction was successful (result replaced dividend $<64:32>$), or a zero if unsuccessful (dividend $<64:32>$ unchanged). After 32 divide steps, the least significant bit of the dividend is aligned at the low end of the high 32 bits of the dividend register (bit $<32:32>$). One final divide step is required to calculate the remainder and least significant quotient bit. This final step shifts the remainder left by one bit however, placing the quotient most significant bit in bit $<32:32>$ (the remainder now occupies bits $<63:33>$ of the dividend register).

When processing very large binary numbers, the correct 33 bit quotient and 31 bit remainder result following 33 divide steps.



EMMY Divide Step Operation

Divide Step proceeds as follows:

- 1) Contents of Reg[AF] | Reg[AF \oplus 1] form the DIVIDEND and Reg[BF] the DIVISOR.
- 2) DIVISOR is subtracted from DIVIDEND<63:32>. If the result is $>=0$, it replaces DIVIDEND<63:32>.
- 3) DIVIDEND is shifted LEFT LOGICALLY by one bit. If result of step 2 was $>=0$, a one is shifted into the low-order bit

position, otherwise if the result of step 2 was <0, a zero is shifted in.

- 4) DIVIDEND<63:32> is returned to Reg[AF], and DIVIDEND<31:0> returned to Reg[AF \oplus 1].

Programming Considerations

- 1) Reg[AF] | Reg[AF \oplus 1] initially contain the 64 bit dividend, while Reg[BF] contains the 32 bit divisor.
- 2) Before entering any divide steps, be sure all operand values are positive. Complement any negative values, remembering which operands were complemented.

Note: dividend must be treated as a 64 bit 2's complement binary number. Sign bit is Reg[AF]<31:31>. It must be complemented as a 64 bit number, not two 31 bit numbers.

- 3) Divisor value must not be zero.
- 4) 33 Divide Steps are required to divide a 64 bit dividend by a 32 bit divisor.

5) After 33 Divide Steps:

Reg[AF]<0:0>|Reg[AF ⊕ 1]<31:0> contains the 33 bit quotient

Reg[AF]<31:1> contains the 31 bit remainder

Since the quotient will rarely be greater than 32 bits, the high order bit can usually be ignored, and Reg[AF] shifted right logically by one bit to right align the remainder in that register.

6) Divide Step is coded as follows:

DIV(AF,BF)

7) Overflow may occur if the number of significant bits in DIVIDEND<63:32> is more than one greater than the number of significant bits in the Divisor. (Quotient requires more than 33 bits). This condition will not be detected by the hardware.

8) For other than general case operands, fewer Divide Steps and some pre- and post-shifting may be used to accomplish division.

1.5 Branching and Conditional

The EMMY Processor contains two conditionally executed instructions, one a T-op, the other an A-op. In each case, an 8-bit mask and 3 modifier bits are specified. The modifier bits are denoted V, C, and S.

The mask and modifier bits dictate a test of either the CCODES or ICODES of R0.

The mask specification is used to select a subset of the specified codes, ie CCODES or ICODES if S=0 or 1 respectively. A code bit is selected for test if its corresponding mask bit is 1. If the C bit is one, the subset of codes is inverted.

The V bit controls the sense of the test. If V=0, the test will be true if any subset bit is true, otherwise the test fails. If V=1, the test will be true if all subset bits are zero, and false otherwise.

In summary, the mask is used to select a subset of the codes specified by S. If C=1, the subset is inverted. The test is true if V=0 and any subset bit (or inverted subset bit if C=1) is one, or if V=1 and all subset bits (or inverted subset bits if C=1) are zero. Otherwise, the test fails.

1.51 T-Machine Conditional

The T-op conditional controls the execution of the A-op contained in bits <17:0>. If the conditional test specified proves true, the A-op is suppressed; if the test proves false, the A-op is executed.

1.52 A-Machine Conditional

The A-machine conditional controls sequencing of microinstructions. A conditional test is specified as well as a 4-bit value, VAL. If the test proves true, then VAL, sign extended to 12 bits, is added to $R0<11:0>$, the microinstruction counter. Forward branches of up to 8 instructions beyond the current instruction ($R0 + 7$), or backward branches of up to 7 instructions prior to the current instruction ($R0 - 8$) are possible.

Programming Considerations

- 1) Conditions which may be detected are "any selected bit one or zero" and "all selected bits one or zero".

2) To build a conditional:

i) set S to specify ICODES (S=1) or CCODES (S=0).

ii) set mask to select desired code bits

iii) for T-op conditional, if you want the A-op executed if any selected bit is one/zero, set V=1, otherwise, if you want the A-op executed if all selected bits are one/zero, set V=0. For the A-op branch, if you wish to branch if any selected bit is one/zero, set V=0, otherwise, if you wish to branch if all selected bits are one/zero, set V=1.

iv) All mask definitions should be referred to the T-side, as described above. The assembler will make any adjustments for "NOT" conditions and A-side branches.

v) If you wish to detect any selected

bit one or all selected bits zero,
set C=0, otherwise, if you wish to
detect any selected bit zero or all
selected bits one, set C=1.

- 2) A conditional mask is defined in EMMYXL by
EQUating a tag to a MASK function specified as follows:

<tag> EQU MASK(<mask>,<v>,<c>,<s>)

where <mask> is an absolute numeric value corresponding to
the desired 8 bit mask, ie 255 ==> B'11111111' ==> select all
bits. <v>,<c>,<s> specify V,C, and S bits and must be either
one or zero. A mask function should not be coded directly in
a conditional, rather EQUATE a tag to the function and use
the tag.

2.0 Crossassembler Operation

This section describes the use of EMMYXL as implemented under IBM VSII. The actual assembler is currently written in ALGOLW, but will eventually be converted to PL360.

2.1 Titles and Comments

Comments are coded by placing a period (".") prior to the first character of text. This causes the assembler to ignore the rest of the input card. This text is printed with the source listing. In addition, blank lines may be included to improve the readability of source listings.

A page eject is indicated by a plus ("+") in column 1. If columns 2 through 61 are not blank, they become the new title and are printed at the top of all subsequent pages (until a new title is defined). If no new title is indicated, the previous one, if any, is printed.

2.2 Location Counter Control

All addresses processed by EMMYXL are EMMY bus addresses.

Thus, micromemory addresses are X'FF0000' through X'FF0FFF', and the host register file addresses X'FF1000' through X'FF1007'. Main store addresses are currently X'000000' through X'003FFF'. The assembler can be used to initialize any location in the EMMY system memory, including the registers and main store.

The assembler location counter can be set using the ORIGIN pseudo op. The argument of the ORG instruction must be a valid bus address or a symbol whose value is a valid bus address. All tags attached to machine instructions are assigned the current value of the location counter, which is, of course, a bus address. When coding a micromemory location as a literal, be sure to code a control store address. (eg-to begin assembly at location X'100', code ORG X'FF0100').

2.3 Listing Options

By default, the assembler will produce a source listing during PASS 2. This listing consists of a title, which contains the language processor identifier and version date, a user supplied title, and page number. The title/page eject card is discussed in section 2.1. Following the title are up to 55 lines of assembled source text. Column 1 contains an error flag, if there was any error in that source statement, or a space, followed

by the location counter value, which is an EMMY bus address. An 8 hexadecimal digit constant follows, representing the assembled object text. The card number and source text complete the line. Comments and assembler directives are printed with blank location counter and object text fields. EQU's are printed with blank location counter fields and the EQUATED value in the object text field.

The source listing may be suppressed by including a "&NOLIST" card anywhere in the source stream. This card is interpreted during PASS1 and will prevent any source listing from being produced.

Portions of a source listing can be suppressed by coding "&NOPRINT" card just prior to the point at which the listing should be suppressed, and placing a "&PRINT" card just prior to the point at which the listing should resume. These cards affect only the printed listing, not the assembled object text.

2.4 Code Generation

When the assembler directive "&CODE" is included in the source text, the assembler produces an object text file consisting of a bus address and text unpacked in ASCII representation of

hexadecimal values. This object file is included at the end of the assembler listing, preceded by the title:

***** EMMY OBJECT LISTING *****

and several ASCII control characters for use in transmitting the object text to the lab's Datapoint terminal. See Uniterm II User's Guide for details.

The default is to generate no object text file. A "&CODE" card must be included if such an output is desired.

See Appendix C for listing formats and object text formats

3.0 EMMY Simulator

The EMMYXL package includes a simulator program, whereby the EMMY code assembled can be tested. Operation of the simulator is the same as the actual EMMY.

When a "&SIM" card is included, and no errors occurred during assembly, the assembled object code is loaded into simulated control store, mainstore and host registers. Microinstructions are fetched from the location contained in R0<11:0>. The starting point for the simulation can be set by ORGing to R0 and defining a constant equal to that address, or as an argument of the END statement.

Simulation ends when the HALT bit becomes set, or an illegal operation is attempted. At this time, a post execution dump of micromemory and the host register file is printed.

When a "&TRACE" card is included, an instruction trace is performed as the simulation progresses. Each instruction cycle, the microinstruction register and host registers are printed in hexadecimal.

Default conditions are NOSIM and NOTRACE

References and Related Material

- 1.0 Hedges, Thomas, EMMY/360 Cross Assembler, Stanford Technical Note #74, December, 1975.
- 2.0 Neuhauser, Charles, Dynamic Microprogrammable Processor (Version III), Stanford Technical Note #65, December, 1975.
- 3.0 Datapoint Corporation, Uniterm2/Uniterm3, 3.1 User's Guide, June 27, 1974, Datapoint Corporation, 9725 Datapoint Dr, San Antonio, Texas 78284.

Appendix A Error Flags

- A **Illegal A-statement**
 - a) missing ")"
 - b) missing ";"
 - c) illegal syntax or operator (eg, missing "=")
 - d) missing expression or illegal "-"

- B **"BLK" statement error**
 - a) <abs> not > 0

- C **Illegal constant**
 - a) Hex constant specifies other than 0-9, A-F
 - b) Octal constant specifies other than 0-7
 - c) Binary constant specifies other than 0-1
 - d) illegal DC

- D
 - a) illegal conditional
 - b) address not > 0 in "ORG"

- E
 - a) illegal END card
 - b) illegal expression

- I **Insert/Extract error**
 - a) specified field(s) can not be assembled into a legal
 literal mask (see L flag)

K illegal "MASK" function

- a) missing "("
- b) <cmask> or <bmask> out of range (not $\geq 0, \leq 255$)
- c) V,C,S not 0 or 1
- d) values not separated by comma (",")

L illegal literal

- a) literal does not specify 16 bit constant zero or one filled on right or left to 32 bits in expanded B-field
-) literal out of range (not $\geq 0, \leq 4095$ in LOAD IMMEDIATE)

M syntax error

- a) composite operator illegally written (eg ":" = written as ": =")

P procedural error

- a) literal out of range (lit not $\geq -8, \leq 7$)
 in branch, pointer mod
- b) illegal conditional in pointer mod

Q undetermined error

R relocatability error

S PASS2 LC not = PASS1 LC - internal error has occurred

T illegal T-statement

- a) illegal syntax
- b) illegal operator
- c) ilegal operand

U undefined symbol

X illegal pointer mod after indirect access

- a) other than CF to DF specified
- b) illegal syntax (ie not "+" or "-")

Appendix B Using EMMYXL

All necessary JCL to use EMMYXL has been included in the catalogued procedure EMMYXL. The input specified must be a Wylbur EDIT format data set containing all text and control cards. This data set must be catalogued. If the name contains qualifiers (or indicies, in SCIP terminology), that is, qualifying name(s) separated by period(s), it must be enclosed in single quotes in the EXEC statement. Additionally, no qualifier may be longer than 8 characters, nor begin with other than an alphabetic character (the same restrictions apply to simple names).

A GROUP and USER must be specified, reflecting the account under which the source data set is stored. The source data set must be catalogued

If the source text contains both upper and lower case characters, an uplow listing may be obtained by coding ",SYSOT=D" in the EXEC statement (note the misspelling of SYSOUT). Only the assembler listing will be printed uplow, so watch for two listings.

examples:

```
// EXEC EMMYXL,SOURCE=myfile,GROUP=gg,USER=uuu  
file accessed is 'WYL.gg.uu myfile'. Listing is upper case only.
```

```
// EXEC EMMYXL,SOURCE=myprog.versl,USER=uuu,GROUP=gg  
  
illegal - SOURCE ia a qualified name (contains special  
character,".") and is not enclosed in single quotes.
```

```
// EXEC EMMYXL,SOURCE=360emu,GROUP=gg,USER=uuu,SYSTOT=d  
  
illegal - SOURCE contains qualifier which begins with numeric.
```

```
// EXEC EMMYXL,SOURCE='text.versl',GROUP=gg,USER=uuu  
  
file accessed is 'WYL.gg.uuu.text.versl', Listing is upper case  
only.
```

The EMMYXL procedure executes two job steps. The first UNPRESSES the EDIT format source and creates a card-image scratch data set, which is PASSED to the second step.

The second job step reads the scratch data set as input to the assembler. This data set is scratched at step termination.

The original source, however, is kept.

BEST AVAILABLE COPY

Appendix C: SAMPLE EMMYXL LISTINGS

```

JOB 1322 ① Procedure Call
2.

***** FMMYXL 370 CROSS ASSEMBLER FÜR STANFORD EMMY
***** SOURCE IS A MYLER EDIT FORMAT DATA SET WHICH
***** CONTAINS SOURCE TEXT AND EMMY CONTROL CARDS.
***** QUALIFIERS MUST BE <=8 CHARACTERS AND BEGIN WITH
***** ALPHABETIC CHARACTER.
***** DATA SET MUST ALSO BE CATALOGED.
***** SYSCOT IS SYSCUT CLASS FOR ASSEMBLER LISTING
***** CONSULTANT: WALTER WALLACH
***** AFFILIATION: DIGITAL SYSTEMS LAB-ELECT.ENGR.
***** TELEPHONE: (4917-0377
***** ADDRESS: EFL 322
***** SUBMITTED: C3/11/76
***** EXPIRES:
***** XWUPIPES EXEC PGM=IDOPDGM
***** AXST:PLIB ED DSN=SYS4.IDOPDCM.LIB,DISP=SHR
***** XSYSPRINT CC SYSOUT=A
***** XXSYSPRINT ED SYSOUT=B
***** XXXI:PUT DC DSN=WYL.8GROLF..EJSR..LSDJURCE,DISP=SHR
***** IEF6531 SUBSTITUTION JCL - CSN=HYL.KUMXF.MULTDIV,DISP=SHR
***** XXXI:PUT ED DSN=SYSDA,SPACE=(3200,130,10),DISP=(NEW,PASS),
***** DCB=LRECL=80,LKSIZE=3200,RECM=FB
***** DD DSN=WLD.D2.N27.CCOMMAND,DISP=SHR
***** XXSYSIN DD FOR MXKU071 UNPRES
***** IEF2361 ALLOC. FOR MXKU071 UNPRES
***** IEF2371 16E ALLOCATED TO SYPRINT
***** IEF2371 D36 ALLOCATED TO STEPLIB
***** IEF2371 D23 ALLOCATED TO SYSPUNCH
***** IEF2371 17C ALLOCATED TO INPUT
***** IEF2371 51F ALLOCATED TO CPUT
***** IEF2371 17C ALLOCATED TO SYSIN
***** IEF11+21 - STEP WAS EXECUTED - CCCC CODE 0000

***** XWEMY EXEC PGN=EMMYL,PAQE=PAGE3,SIZE=500,TIME=30*
***** XXST:PLIB ED DSN=WYL.C2.N27.EFWYXL.JDSLIB,DISP=SHR
***** XSYSPRINT DD SYSOUT=EJSOT
***** IEF6531 SUBSTITUTION JCL - SYSOUT=A
***** UNIT=SYSDA,SPACE=(3404,1125,25),DISP=(NEW,DELETE),
***** XXXI:PUT ED CCA=1,RECFCM=VRS,BLKSIZE=39,44
***** XXFTDSFO01 DD DONAM=SYSIN
***** XXSYSIN DD UNPRES.CLPUT,DISP=(OLD,DELETE)
***** // I-F2361 ALLCC. FOR MXFKL071 EMMY
***** IEF2371 17F ALLOCATED TO STEPLIB
***** IEF2371 C36 ALLOCATED TO SYSPUNCH
***** IEF2371 51D ALLOCATED TO FT10FO01
***** IEF2371 51F ALLOCATED TO FTC5FO01
***** IEC1301 SYSIN DD STATEMENT MISSING
***** SEC1301 SYSIN DD STATEMENT MISSING
***** SEC1421 C-STEP WAS EXECUTED T CCCC CODE 0000
***** //
```

③ Assemble, then scratch card image file

EMMY X L 3/28/76

EMMYXL EXAMPLE 1 FIXED POINT MULTIPLY

1 (3)

①
② ECOCE
③ ESIW
④ ETRACE
⑤
⑥ •REGISTER DEFINITIONS
⑦
⑧ MAR EQU R0
⑨ XR EQU R2
⑩ P EQU R4
⑪ Q EQU R5
⑫ S EQU R7
⑬ .DIVIDEND HIGH HALF
⑭ .DIVIDEND LOW HALF
⑮ .DIVISOR
⑯ .P+Q ALSO FORM THE PRODUCT REGISTER (64 BITS)
⑰ .S IS ALSO THE MULTIPLICAND REGISTER, Q THE MULTIPLIER
⑱ .START AT LOCATION 0
⑲ X'FF0000'
⑳ JRG
⑳ MUL:
⑳ .FIXED POINT MULTIPLY 32 BIT OPERANDS
㉑ .OPERANDS ASSUMED IN REGISTERS Q (MULTIPLIER) AND S (MULTIPLI•ND)
㉒ .CLEAR PRODUCT<63:32> SET ITERATION CTR
㉓ P := 0 ; XR = 32
㉔ MUS(P,S) ; DEC XR (~<0 => MAR-1) •33 ITERATIONS
㉕ 27
㉖ 28
㉗ 29
㉘ 30
㉙ 31
㉚ 32
㉛ 33
㉜ 34
㉝ 35
㉞ 36
㉟ 37
㉟ 38
㉟ 39
㉟ 40
㉟ 41
㉟ 42
㉟ 43
㉟ 44
㉟ 45
㉟ 46
㉟ 47
㉟ 48
㉟ 49
㉟ 50
㉟ 51
㉟ 52
㉟ 53
㉟ 54
㉟ 55
㉟ 56
㉟ 57
㉟ 58
㉟ 59
㉟ 60
㉟ 61
㉟ 62
㉟ 63
㉟ 64
㉟ 65
㉟ 66
㉟ 67
㉟ 68
㉟ 69
㉟ 70
㉟ 71
㉟ 72
㉟ 73
㉟ 74
㉟ 75
㉟ 76
㉟ 77
㉟ 78
㉟ 79
㉟ 80
㉟ 81
㉟ 82
㉟ 83
㉟ 84
㉟ 85
㉟ 86
㉟ 87
㉟ 88
㉟ 89
㉟ 90
㉟ 91
㉟ 92
㉟ 93
㉟ 94
㉟ 95
㉟ 96
㉟ 97
㉟ 98
㉟ 99
㉟ 100
㉟ 101
㉟ 102
㉟ 103
㉟ 104
㉟ 105
㉟ 106
㉟ 107
㉟ 108
㉟ 109
㉟ 110
㉟ 111
㉟ 112
㉟ 113
㉟ 114
㉟ 115
㉟ 116
㉟ 117
㉟ 118
㉟ 119
㉟ 120
㉟ 121
㉟ 122
㉟ 123
㉟ 124
㉟ 125
㉟ 126
㉟ 127
㉟ 128
㉟ 129
㉟ 130
㉟ 131
㉟ 132
㉟ 133
㉟ 134
㉟ 135
㉟ 136
㉟ 137
㉟ 138
㉟ 139
㉟ 140
㉟ 141
㉟ 142
㉟ 143
㉟ 144
㉟ 145
㉟ 146
㉟ 147
㉟ 148
㉟ 149
㉟ 150
㉟ 151
㉟ 152
㉟ 153
㉟ 154
㉟ 155
㉟ 156
㉟ 157
㉟ 158
㉟ 159
㉟ 160
㉟ 161
㉟ 162
㉟ 163
㉟ 164
㉟ 165
㉟ 166
㉟ 167
㉟ 168
㉟ 169
㉟ 170
㉟ 171
㉟ 172
㉟ 173
㉟ 174
㉟ 175
㉟ 176
㉟ 177
㉟ 178
㉟ 179
㉟ 180
㉟ 181
㉟ 182
㉟ 183
㉟ 184
㉟ 185
㉟ 186
㉟ 187
㉟ 188
㉟ 189
㉟ 190
㉟ 191
㉟ 192
㉟ 193
㉟ 194
㉟ 195
㉟ 196
㉟ 197
㉟ 198
㉟ 199
㉟ 200
㉟ 201
㉟ 202
㉟ 203
㉟ 204
㉟ 205
㉟ 206
㉟ 207
㉟ 208
㉟ 209
㉟ 210
㉟ 211
㉟ 212
㉟ 213
㉟ 214
㉟ 215
㉟ 216
㉟ 217
㉟ 218
㉟ 219
㉟ 220
㉟ 221
㉟ 222
㉟ 223
㉟ 224
㉟ 225
㉟ 226
㉟ 227
㉟ 228
㉟ 229
㉟ 230
㉟ 231
㉟ 232
㉟ 233
㉟ 234
㉟ 235
㉟ 236
㉟ 237
㉟ 238
㉟ 239
㉟ 240
㉟ 241
㉟ 242
㉟ 243
㉟ 244
㉟ 245
㉟ 246
㉟ 247
㉟ 248
㉟ 249
㉟ 250
㉟ 251
㉟ 252
㉟ 253
㉟ 254
㉟ 255
㉟ 256
㉟ 257
㉟ 258
㉟ 259
㉟ 260
㉟ 261
㉟ 262
㉟ 263
㉟ 264
㉟ 265
㉟ 266
㉟ 267
㉟ 268
㉟ 269
㉟ 270
㉟ 271
㉟ 272
㉟ 273
㉟ 274
㉟ 275
㉟ 276
㉟ 277
㉟ 278
㉟ 279
㉟ 280
㉟ 281
㉟ 282
㉟ 283
㉟ 284
㉟ 285
㉟ 286
㉟ 287
㉟ 288
㉟ 289
㉟ 290
㉟ 291
㉟ 292
㉟ 293
㉟ 294
㉟ 295
㉟ 296
㉟ 297
㉟ 298
㉟ 299
㉟ 300
㉟ 301
㉟ 302
㉟ 303
㉟ 304
㉟ 305
㉟ 306
㉟ 307
㉟ 308
㉟ 309
㉟ 310
㉟ 311
㉟ 312
㉟ 313
㉟ 314
㉟ 315
㉟ 316
㉟ 317
㉟ 318
㉟ 319
㉟ 320
㉟ 321
㉟ 322
㉟ 323
㉟ 324
㉟ 325
㉟ 326
㉟ 327
㉟ 328
㉟ 329
㉟ 330
㉟ 331
㉟ 332
㉟ 333
㉟ 334
㉟ 335
㉟ 336
㉟ 337
㉟ 338
㉟ 339
㉟ 340
㉟ 341
㉟ 342
㉟ 343
㉟ 344
㉟ 345
㉟ 346
㉟ 347
㉟ 348
㉟ 349
㉟ 350
㉟ 351
㉟ 352
㉟ 353
㉟ 354
㉟ 355
㉟ 356
㉟ 357
㉟ 358
㉟ 359
㉟ 360
㉟ 361
㉟ 362
㉟ 363
㉟ 364
㉟ 365
㉟ 366
㉟ 367
㉟ 368
㉟ 369
㉟ 370
㉟ 371
㉟ 372
㉟ 373
㉟ 374
㉟ 375
㉟ 376
㉟ 377
㉟ 378
㉟ 379
㉟ 380
㉟ 381
㉟ 382
㉟ 383
㉟ 384
㉟ 385
㉟ 386
㉟ 387
㉟ 388
㉟ 389
㉟ 390
㉟ 391
㉟ 392
㉟ 393
㉟ 394
㉟ 395
㉟ 396
㉟ 397
㉟ 398
㉟ 399
㉟ 400
㉟ 401
㉟ 402
㉟ 403
㉟ 404
㉟ 405
㉟ 406
㉟ 407
㉟ 408
㉟ 409
㉟ 410
㉟ 411
㉟ 412
㉟ 413
㉟ 414
㉟ 415
㉟ 416
㉟ 417
㉟ 418
㉟ 419
㉟ 420
㉟ 421
㉟ 422
㉟ 423
㉟ 424
㉟ 425
㉟ 426
㉟ 427
㉟ 428
㉟ 429
㉟ 430
㉟ 431
㉟ 432
㉟ 433
㉟ 434
㉟ 435
㉟ 436
㉟ 437
㉟ 438
㉟ 439
㉟ 440
㉟ 441
㉟ 442
㉟ 443
㉟ 444
㉟ 445
㉟ 446
㉟ 447
㉟ 448
㉟ 449
㉟ 450
㉟ 451
㉟ 452
㉟ 453
㉟ 454
㉟ 455
㉟ 456
㉟ 457
㉟ 458
㉟ 459
㉟ 460
㉟ 461
㉟ 462
㉟ 463
㉟ 464
㉟ 465
㉟ 466
㉟ 467
㉟ 468
㉟ 469
㉟ 470
㉟ 471
㉟ 472
㉟ 473
㉟ 474
㉟ 475
㉟ 476
㉟ 477
㉟ 478
㉟ 479
㉟ 480
㉟ 481
㉟ 482
㉟ 483
㉟ 484
㉟ 485
㉟ 486
㉟ 487
㉟ 488
㉟ 489
㉟ 490
㉟ 491
㉟ 492
㉟ 493
㉟ 494
㉟ 495
㉟ 496
㉟ 497
㉟ 498
㉟ 499
㉟ 500
㉟ 501
㉟ 502
㉟ 503
㉟ 504
㉟ 505
㉟ 506
㉟ 507
㉟ 508
㉟ 509
㉟ 510
㉟ 511
㉟ 512
㉟ 513
㉟ 514
㉟ 515
㉟ 516
㉟ 517
㉟ 518
㉟ 519
㉟ 520
㉟ 521
㉟ 522
㉟ 523
㉟ 524
㉟ 525
㉟ 526
㉟ 527
㉟ 528
㉟ 529
㉟ 530
㉟ 531
㉟ 532
㉟ 533
㉟ 534
㉟ 535
㉟ 536
㉟ 537
㉟ 538
㉟ 539
㉟ 540
㉟ 541
㉟ 542
㉟ 543
㉟ 544
㉟ 545
㉟ 546
㉟ 547
㉟ 548
㉟ 549
㉟ 550
㉟ 551
㉟ 552
㉟ 553
㉟ 554
㉟ 555
㉟ 556
㉟ 557
㉟ 558
㉟ 559
㉟ 560
㉟ 561
㉟ 562
㉟ 563
㉟ 564
㉟ 565
㉟ 566
㉟ 567
㉟ 568
㉟ 569
㉟ 570
㉟ 571
㉟ 572
㉟ 573
㉟ 574
㉟ 575
㉟ 576
㉟ 577
㉟ 578
㉟ 579
㉟ 580
㉟ 581
㉟ 582
㉟ 583
㉟ 584
㉟ 585
㉟ 586
㉟ 587
㉟ 588
㉟ 589
㉟ 590
㉟ 591
㉟ 592
㉟ 593
㉟ 594
㉟ 595
㉟ 596
㉟ 597
㉟ 598
㉟ 599
㉟ 600
㉟ 601
㉟ 602
㉟ 603
㉟ 604
㉟ 605
㉟ 606
㉟ 607
㉟ 608
㉟ 609
㉟ 610
㉟ 611
㉟ 612
㉟ 613
㉟ 614
㉟ 615
㉟ 616
㉟ 617
㉟ 618
㉟ 619
㉟ 620
㉟ 621
㉟ 622
㉟ 623
㉟ 624
㉟ 625
㉟ 626
㉟ 627
㉟ 628
㉟ 629
㉟ 630
㉟ 631
㉟ 632
㉟ 633
㉟ 634
㉟ 635
㉟ 636
㉟ 637
㉟ 638
㉟ 639
㉟ 640
㉟ 641
㉟ 642
㉟ 643
㉟ 644
㉟ 645
㉟ 646
㉟ 647
㉟ 648
㉟ 649
㉟ 650
㉟ 651
㉟ 652
㉟ 653
㉟ 654
㉟ 655
㉟ 656
㉟ 657
㉟ 658
㉟ 659
㉟ 660
㉟ 661
㉟ 662
㉟ 663
㉟ 664
㉟ 665
㉟ 666
㉟ 667
㉟ 668
㉟ 669
㉟ 670
㉟ 671
㉟ 672
㉟ 673
㉟ 674
㉟ 675
㉟ 676
㉟ 677
㉟ 678
㉟ 679
㉟ 680
㉟ 681
㉟ 682
㉟ 683
㉟ 684
㉟ 685
㉟ 686
㉟ 687
㉟ 688
㉟ 689
㉟ 690
㉟ 691
㉟ 692
㉟ 693
㉟ 694
㉟ 695
㉟ 696
㉟ 697
㉟ 698
㉟ 699
㉟ 700
㉟ 701
㉟ 702
㉟ 703
㉟ 704
㉟ 705
㉟ 706
㉟ 707
㉟ 708
㉟ 709
㉟ 710
㉟ 711
㉟ 712
㉟ 713
㉟ 714
㉟ 715
㉟ 716
㉟ 717
㉟ 718
㉟ 719
㉟ 720
㉟ 721
㉟ 722
㉟ 723
㉟ 724
㉟ 725
㉟ 726
㉟ 727
㉟ 728
㉟ 729
㉟ 730
㉟ 731
㉟ 732
㉟ 733
㉟ 734
㉟ 735
㉟ 736
㉟ 737
㉟ 738
㉟ 739
㉟ 740
㉟ 741
㉟ 742
㉟ 743
㉟ 744
㉟ 745
㉟ 746
㉟ 747
㉟ 748
㉟ 749
㉟ 750
㉟ 751
㉟ 752
㉟ 753
㉟ 754
㉟ 755
㉟ 756
㉟ 757
㉟ 758
㉟ 759
㉟ 760
㉟ 761
㉟ 762
㉟ 763
㉟ 764
㉟ 765
㉟ 766
㉟ 767
㉟ 768
㉟ 769
㉟ 770
㉟ 771
㉟ 772
㉟ 773
㉟ 774
㉟ 775
㉟ 776
㉟ 777
㉟ 778
㉟ 779
㉟ 780
㉟ 781
㉟ 782
㉟ 783
㉟ 784
㉟ 785
㉟ 786
㉟ 787
㉟ 788
㉟ 789
㉟ 790
㉟ 791
㉟ 792
㉟ 793
㉟ 794
㉟ 795
㉟ 796
㉟ 797
㉟ 798
㉟ 799
㉟ 800
㉟ 801
㉟ 802
㉟ 803
㉟ 804
㉟ 805
㉟ 806
㉟ 807
㉟ 808
㉟ 809
㉟ 810
㉟ 811
㉟ 812
㉟ 813
㉟ 814
㉟ 815
㉟ 816
㉟ 817
㉟ 818
㉟ 819
㉟ 820
㉟ 821
㉟ 822
㉟ 823
㉟ 824
㉟ 825
㉟ 826
㉟ 827
㉟ 828
㉟ 829
㉟ 830
㉟ 831
㉟ 832
㉟ 833
㉟ 834
㉟ 835
㉟ 836
㉟ 837
㉟ 838
㉟ 839
㉟ 840
㉟ 841
㉟ 842
㉟ 843
㉟ 844
㉟ 845
㉟ 846
㉟ 847
㉟ 848
㉟ 849
㉟ 850
㉟ 851
㉟ 852
㉟ 853
㉟ 854
㉟ 855
㉟ 856
㉟ 857
㉟ 858
㉟ 859
㉟ 860
㉟ 861
㉟ 862
㉟ 863
㉟ 864
㉟ 865
㉟ 866
㉟ 867
㉟ 868
㉟ 869
㉟ 870
㉟ 871
㉟ 872
㉟ 873
㉟ 874
㉟ 875
㉟ 876
㉟ 877
㉟ 878
㉟ 879
㉟ 880
㉟ 881
㉟ 882
㉟ 883
㉟ 884
㉟ 885
㉟ 886
㉟ 887
㉟ 888
㉟ 889
㉟ 890
㉟ 891
㉟ 892
㉟ 893
㉟ 894
㉟ 895
㉟ 896
㉟ 897
㉟ 898
㉟ 899
㉟ 900
㉟ 901
㉟ 902
㉟ 903
㉟ 904
㉟ 905
㉟ 906
㉟ 907
㉟ 908
㉟ 909
㉟ 910
㉟ 911
㉟ 912
㉟ 913
㉟ 914
㉟ 915
㉟ 916
㉟ 917
㉟ 918
㉟ 919
㉟ 920
㉟ 921
㉟ 922
㉟ 923
㉟ 924
㉟ 925
㉟ 926
㉟ 927
㉟ 928
㉟ 929
㉟ 930
㉟ 931
㉟ 932
㉟ 933
㉟ 934
㉟ 935
㉟ 936
㉟ 937
㉟ 938
㉟ 939
㉟ 940
㉟ 941
㉟ 942
㉟ 943
㉟ 944
㉟ 945
㉟ 946
㉟ 947
㉟ 948
㉟ 949
㉟ 950
㉟ 951
㉟ 952
㉟ 953
㉟ 954
㉟ 955
㉟ 956
㉟ 957
㉟ 958
㉟ 959
㉟ 960
㉟ 961
㉟ 962
㉟ 963
㉟ 964
㉟ 965
㉟ 966
㉟ 967
㉟ 968
㉟ 969
㉟ 970
㉟ 971
㉟ 972
㉟ 973
㉟ 974
㉟ 975
㉟ 976
㉟ 977
㉟ 978
㉟ 979
㉟ 980
㉟ 981
㉟ 982
㉟ 983
㉟ 984
㉟ 985
㉟ 986
㉟ 987
㉟ 988
㉟ 989
㉟ 990
㉟ 991
㉟ 992
㉟ 993
㉟ 994
㉟ 995
㉟ 996
㉟ 997
㉟ 998
㉟ 999
㉟ 1000

- 1 Language Processor Identifier and Version Date
- 2 User Title
- 3 Flag (none in this assembly)
- 4 Location (bus address)
- 5 Object Code (or value, in the case of EQU, Mask)
- 6 Input Card Number
- 7 Source Text

30 .THE DIVIDE ALGORITHM WILL WORK WITH POSITIVE VALUES ONLY.
 31 .THEREFORE, WE MUST COMPLEMENT NEGATIVE VALUES AND KEEP TRACK OF
 32 WHICH VALUES WERE COMPLEMENTED. THE REMAINDER MUST BE COMPLEMENTED
 33 .IF THE DIVIDEND WAS NEGATIVE, AND THE QUOTIENT COMPLEMENTED IF
 34 EITHER THE DIVIDEND OR DIVISOR, BUT NOT BOTH, WERE NEGATIVE.
 35 .ICCODE BIT <1:1> WILL INDICATE REMAINDER TO BE COMPLEMENTED, AND
 36 .ICCODE BIT <0:0> INDICATE QUOTIENT TO BE COMPLEMENTED.
 37 .ICCODE BIT <0:0> INDICATE QUOTIENT TO BE COMPLEMENTED.

000000011	39	CCMPREM	EQU	MASK(2,0,0,1)	.FLAG FOR REMAINDER
000000039	40	NEG7	EQU	MASK(1,0,2,1)	.FLAG FOR QUOTIENT
FF0002	41				
FF0003	42				
FF0004	43				
FFJ005	44				
FF0006	45	S := S	; ! NOT ZERO => MAR+1)	.IS DIVISOR ZERO?	
FF0007	46	PAR := MAR	; ! NOT ZERO => MAR+1)	.YES - HALT	
FF0008	47				
FF0009	48				
FF000A	49	{ POSITIVE => ; MAR = *++ 3)			.IS DIVISOR NEGATIVE?
FF000B	50	MAR := MAR XOR X'00000000'			* YES - SET QUOTIENT FLAG
FF000C	51	S := ~S	; INC S		* COMPLEMENT - IGNORE OLF
FFJ00A	52	P := P	; !NOT NEGATIVE => MAR+5) .IS DIVIDEND NEG?		
FFQ00B	53	MAR := MAR XOR X'00000000'			* YES INVERT QUOTIENT FLAG
FFQ00C	54				* SET REMAINDER FLAG
FF000D	55	Q := ~Q			
FF000E	56	P := ~P			* ONES COMPL
FF000F	57	Q := Q+1			* INCREMENT 64 BIT DIVIDEND
FF0010	58	P := P+C+ 0			
FF0011	59				
FF0012	60				
FF0013	61	XR := 32			
FF0014	62				
FF0015	63	DIV(P,S) ; DEC XR (&<0 => MAR-1) .33 ITERATIONS			
FF0016	64	P >> 1			* TRUNCATE QUOTIENT TO 32 BITS
FF0017	65				* AND LEAVE REMAINDER RIGHT JUST
FF0018	66				
FF0019	67	{ COMPLEMENT => ; MAR=**+2) .MUST WE COMPLEMENT REM?			
FF001A	68	P := ~P ; INC P			* YES - IGNORE OLF
FF001B	69				
FF001C	70	{ NEGT => ; MAR=**+2) .HOW ABOUT QUOTIENT?			
FF001D	71	Q := ~Q ; INC Q			* YES - IGNORE OLF
FF001E	72				
FF001F	73	{ NUM Q COUNTAINS QUOTIENT AND P CONTAINS REMAINDER			
FF0020	74				
FF0021	75	MAR := MAR OR X'00000000'			
FF0022	76				* HALT
FF0023	77	DIVISOR: DC 1,			
FF0024	78	DIVISOR: DC --1			
FF0025	79	DIVISOR: DC --21*			
FF0026	80	DIVISOR: DC --21*			

BEST AVAILABLE COPY

BEST AVAILABLE COPY

PAGE 3

HOST REGISTER INITIALIZATION	
EMMY X L	3/28/76
	82 NOW INITIALIZE HOST REGISTERS
FF1000	83
FF1000	84 JRG X'FF1000'
FF1001	85 DC X'00000000'
FF1003	86 BLK 3
FF1004	87 DC 0
FF1005	88 DC 45
FF1006	89 DC 0
FF1007	90 DC 298
C00000	91 END

•RO START AT LOCATION 0
 •UNUSED REGISTERS
 •O-REG - MULTIPLIER
 •R-REG NOT USED
 •S-REG MULTPLICAND

1 Reset Location Counter to Host Register File

BEST AVAILABLE COPY

PAGE 4

SYMBOL TABLE LISTING

3/28/76

EMMY X L

	MASK	0000000C	EMMY Symbol Table
	MASK	00J000104	1 Tag or Name
CARRY	MASK	00J000111	2 Identifier Type
CMPREL	MASK	0JFFF0019	3 value (in case of EQU, MASK, etc) or location (in case of Tag)
CIVID1	SYM	0JFFF001A	
CIVID2	SYM	0JFFF001B	
DIVISOR	SYM	0JFFF0018	
HIGH	MASK	0J00C034	
LOW	MASK	0J00C044	
MAR	REG	00000000	
MULT	SYM	0JFFC000	
NEGATIVE	MASK	0000C084	
N-NCT	MASK	0000C009	
ODD	MASK	0000J014	
OVERFLOW	MASK	0000C602	
POSITIVE	REG	0000C004	
POSITIVE	MASK	0J00J00AO	
S	REG	00000005	
RO	REG	0000C000	
R1	REG	0000C001	
R2	REG	0000C002	
R3	REG	0J00C033	
P4	REG	0J00C004	
R5	REG	0000C004	
R6	REG	00J00006	
R7	REG	0000C007	
S	MASK	0J00J0007	
SAME	REG	0000C024	
XR	MASK	0000C002	
ZERO	MASK	0J00C600	(2)
			(1)

Appendix D Sample Object Code and Object File Format

EMMYXL will produce an object text file at the end of the source listing if the assembler directive "&CODE" was included in the source. This file consists of EMMY bus addresses and object text, unpacked into ASCII representation of hexadecimal values. Thus, the hexadecimal value X'9C' would appear as two ASCII characters "9" and "C" printed as part of a string of such characters. These must be packed and converted to their binary equivalents before loading into the EMMY memory system. A loader program is incorporated in the console debug program which accepts this object format, performs the necessary conversions, and loads the text into the EMMY system.

Object text records consist of a six character address, preceded by an address identifier character ("0"). This is followed by up to 64 characters of object text. Each object text record contains an address.

The object text is preceded by two control characters, which are used to control the tape unit of the Datapoint 2200. The ASCII "ENQ" is interpreted by the Uniterm (r) program as "enable cassette" (X'2D'), and enables data to be written to the front cassette unit. The second control character, "DC-2" or "TAPE-ON" (X'12') actually begins writing data to the cassette.

Following the object text appear two control characters which are the complement of the first two. A "DC-4" or "TAPE-OFF" stops writing data to the cassette and writes the last record (purges the Datapoint buffer), and an "EOT" (X'37') disables the cassette unit.

Object File Format

```
"ENQ"    (X'2D')
"TAPE-ON" (X'12')
      any number of object text records
"TAPE-OFF" (X'3C')
"EOT" (X'37')
```

This follows the title "***** EMMY OBJECT CODE *****" in the source listing.

BEST AVAILABLE COPY

BEST AVAILABLE COPY

Appendix E: SAMPLE SIMULATION RUN

INSTRUCTION TRACE

BEST AVAILABLE COPY

NORMAL END OF SIMULATION

34

- | | | | |
|-----|--|---------------------------|--|
| 1 | 33 Multiply Steps | | |
| 2 | Result = X'3462' = 13,410 = 45*298 | | |
| 3,4 | Load values for Division
Complement Dividend and set flag in RO | | |
| 5 | 33 Divide Steps | | |
| 6 | Complement Quotient and Remainder | answer = -16 remainder -6 | |

BEST AVAILABLE COPY

*** EMU MEMORY DUMP ***
R0= 8003801E P1= 00300003 R2= FFFFFFFF R3= 00000000 R4= FFFFFFFA R5= FFFFFFF0 R6= 00000000 R7= 00000000
000000 0C13A020 6FF220BF 6D1C1013 6D01C019 6D01DD01A UAF08000 C283800A * OR ?H FM3 H P 1 2 B *
000008 1902C001 00FE7000 0A500845 19020003 0056806J 00928060 32140001 * * -P E 6 2 3 **
000010 1AC60020 6CF220BF 52100001 CU478015 00924000 CO278017 00B65000 1B008000 * LR ?R * 2 G 2 a * 6 P
000018 00000000 FFFFFFFF FFFFFFFF2A FBFBFBFB FBFBFBFB * FBFBFBFB *** END OF DUMP ***

000.78 SECONDS IN EXECUTION

Appendix F: CONTROL CARD SUMMARY

- <text> - period followed by text - comments ignored by EMMYXL
- + [<text>] - plus sign in column 1 - page eject text (if any) replaces user title at head of page
- ORG {<tag>
<abs>} - set location counter - <abs> must be EMMY bus address
- &NOLIST - suppress entire source listing
- &NOPRINT - turn off printing of that portion of source following the &NOPRINT card
- &PRINT - resume printing of all source text
- &CODE - produce EMMY object code
- &SIM - simulate assembled EMMY program if no errors occurred during assembly
- &TRACE - produce instruction trace during simulation

Note: all "&" control cards and title card must begin in column 1.

**ED
78**

